帰納論理プログラミング

AUTHOR(S):

久木田, 水生

CITATION:

久木田, 水生. 帰納論理プログラミング. 哲学論叢 2006, 33: 103-113

ISSUE DATE:

2006

URL:

http://hdl.handle.net/2433/48857

帰納論理プログラミング

久木田　水生

現代論理学は主として演繹的推論を特徴付けるための理論として発展し、そして大きな成果を残してきた。一方で帰納的推論を体系化するという試みは、全く行なわれなかったわけではないが、理論の面でも実践の面でも、それほど重要な成果を生み出しては来なかった。しかしながら最近になって、記号論理学が、帰納的推論を機械的に遂行するシステムを構築するための理論的な基盤において応用され、そのようなシステムのあるものは実践の上でもかなりの成果を挙げている。そのひとつが帰納論理プログラミング（inductive logic programming; ILP）という人工知能の一分野である。本稿では主に論理学との関わりから ILP の発展を概観したい。

1. 論理の機械化

それが世に登場して以来――あるいは登場する以前からも――コンピューターや人工知能を恐れ、批判する人々が常に存在していた。コンピューターや人工知能の利用が広まるにつれ、彼らはそれが決して人間には及ばない理由を考え出そうと躍起になってきた。どこまでも楽天家で科学の善性を疑わない作家、アイザック・アシモフはこのような傾向を「フランケンシュタイン・コンプレックス」と呼び揶揄した。コンピューターが人間に及ばないとする人々の基本的な論調は次のごとくである。コンピューターは人間から明確な指示を与えられて初めて機能する。従って、決まった規則に従って遂行される課題においてはコンピューターが人間を超えることも可能であるが、創造性や直観の必要とされる課題においては、コンピューターは人間に及ばない。

この主張はもっともだが、しかしかつては不可能と考えられた多くの課題が、実際にコンピューターによって遂行されるようになってきていることもまた事実である。これは人間が直観的に行なっているように思われる判断の背後にあるルール、メカニズムを明示化することによって可能になる。このようなことは算術的計算の領域では古くから行なわれてきた。計算とは与えられた入力に対して決まった結果を出力するための手続きに他ならない。そしてその手続きが明確なルールに従っているのならばそれを機械化するという発想も自然なものである。算盤について考えてみればよい。算盤を使えば与えられた数に対して決まった手順に従って玉を動かしていれば誰にでも――その手続きの意味を知らない

人間にすら――正しい結果が出せる。それどころか、決まったルールに従って一定の動作をさせられるシステムであれば、人間でなくてもこの課題を遂行することが出来るのである。

産業革命が社会を大きく変革しつつあった 19 世紀初頭、ちょうどメアリー・シェリーの『フランケンシュタイン』が世間を騒がしていた時代、バベッジは解析の計算を行なうかなり複雑な機械を考えていた。この機械は当時の技術的な限界により、実際に作られることはなかったのだが、彼は多くの人々から「その機械が実現したとして、それは思考していると言えるのか？」という質問を受けたという。同じ頃、論理的推論を自動的に行なう機械を考案し、実際に作ったのがジェヴォンズである。彼はブールによって考えられた代数的論理計算を自動的に行なう機械を発明した。ただしあらゆる論理的な推論を行なうには、ハードウェアの面でも、実現されるべき理論の面でも、彼の時代には十分なものは作られていなかった。

論理的な推論を機械的に行なうことを可能にする、理論上の進展は 19 世紀から 20 世紀にかけて、ペアノやフレーゲらによってもたらされた。彼らは数学の基礎にある論理的推論を公理体系として定式化することに成功した。彼らの目的は、すべての正しい推論を、自明な公理に論理的（演繹的）な推論規則を適用した結果として、明確に特徴付けることであった。

1930 年代には論理的推論を特徴付けるための、フレーゲらとは異なる二つのアプローチが登場した。一つはタルスキのモデル理論であり、もう一つはゲンツェンの自然演繹である。タルスキは文（の集合）に関する「解釈」と「充足」という概念を明確に定義し、それに基づいて、ある文の集合Γを充足する任意の解釈が文の集合Δもまた充足するならば、ΔはΓからの論理的な帰結である、とした。これに対してゲンツェンは、記号列に対する変換の規則を規定し、ある与えられた記号列Γに対して、それらの規則を繰り返し適用することによって導出される記号列Δは、Γからの論理的帰結である、という形で特徴づけを行なった(Israel, 1993)。

1960 年代、J. A. ロビンソンは、タルスキの充足可能性に基づくアプローチと、ゲンツェンの導出可能性に基づくアプローチの両方を利用した、節形式の述語論理の体系を立て、機械による述語論理の定理の証明のための手続きを与えた。

1970 年代以降には、ロビンソンの成果に基づき、論理的・演繹的な推論だけではなく、与えられたデータの一般化、すなわち帰納的な推論さえも自動化するという試みが行なわれた。その一つのアプローチが ILP である。この試みはコンピューターのハードウェアの面での発展と相俟って、理論的な面のみならず、実用的な面でも大きな成功を収めてきた。

たとえばILPを利用したある機械学習システムは、エキスパート・システムの構築の際に、人間の専門家が下す判断の例から、彼らが（明示的に定式化されていない仕方で）従っているルールを抽出することに成功している(Gillies, 1996)。この研究の実用面における成功は、機械はもはや人間から与えられる指示を待つだけの存在ではなく、自らが従うべきルールを経験から抽出することが出来る、ということを意味している。

2. 論理プログラミングと導出原理

ILPはPrologというプログラミング言語を利用している。Prologの大きな特徴は、プログラムが一階述語論理の式によって書かれているということと、導出原理という推論規則を使って与えられた目標を達成しようとするということである（Prologの名前はフランス語の「Programmation en Logique」に由来している）。ここではPrologのプログラムと導出原理について簡単に説明しよう。

Prologはいわゆる「宣言型プログラミング」の一種であり、与えられた課題に対して、その課題を満足する対象を探索するという仕方で機能する。Prologは一階述語論理の論理式（より詳しくは、ホーン節（Horn clause）と呼ばれる特殊な形式の論理式。後述）のリストによってプログラムが書かれるために、特に論理プログラミングと呼ばれている。

論理プログラミングのアプローチは、論理的な推論を一定のルールに従った機械的な手続きにする、というフレーゲのアイディアから始まっている。そしてフレーゲ以後、エルブラン、J. A. ロビンソンなどによって、実際にコンピューターによって一階述語論理の定理を自動的に証明するためのシステムが開発された。ロビンソンは「導出原理（the resolution principle）」という推論規則を適用することで、ある節集合が充足不可能ならば必ず有限の手続きでその反証を構成することができるシステムを考案した(Robinson, 1965)。そしてこのシステムを利用してPrologは開発されている。Prologのプログラムはホーン節をリストアップすることで書かれている。Prologは与えられた質問（一階述語論理の文の形をしている）に対して、プログラムの中を探索して答を返す。

例を挙げよう。次はPrologのプログラムである。

```
grandparent( X, Y) :- parent( X, Z), parent( Z, Y).
parent( john, bob).
parent( john, beth).
parent( jane, bob).
parent( beth, tom).
```

parent(bob, paul).

「grandparent(X, Y) :- parent(X, Z), parent(Z, Y).」は通常の表記では、「∀x∀y[grandparent(x, y)∨ ¬ parent (x, z)∨ ¬ parent(z, y)]」と表わすことが出来る。ホーン節はこのように、一つの正のリテラルと、任意個（0 個でも良い）の負のリテラルを選言結合子∨で結び、すべての変数に全称量化を掛けた冠頭標準形の文と等しいものと考えることが出来る(1)。ホーン節の正のリテラルを、その節の頭部という。それ以外の部分を本体という。

このプログラムの一番目の節のように複数のリテラルから構成される節をルールという。一方一つのリテラルからなる節を事実と呼ぶ。このプログラムは grandparent という述語に関するルールの宣言と、parent という述語と john、bob、beth、jane、tom、paul という個体（アトムと呼ばれる）に関する事実の宣言からなっている

Prolog は論理式によって表現された質問（ゴール節という）に対して答の探索を行なう。例えば

?- parent(john, bob).

と質問すると Prolog は、この事実がプログラム中で宣言されているため、「yes」と答える。

?- parent(bob, john).

と質問すると、このような事実はプログラム中に宣言されていないため、「no」と答える。また

?- grandparent(john, paul).

と質問すると、Prolog は、プログラムの一行目のルールの頭部が parent(X, Y) になっているので、この節の X に john を、Y に paul を入れて、同じ節の二番目のリテラル、parent(john, Z) が成立するような Z を探索する。まず beth が見つかるので、Z に beth をいれて三番目のリテラルの検討に進む（この過程をバックトラックという）。parent(beth, paul) は成立していないのでこの探索は失敗である。そこで Prolog は二番目のリテラルに戻って別な Z の候補を探す。parent(john, bob) が成立しているので、Z に paul を入れて三番目のリテラルに進む。すると parent(bob, paul) が事実として宣言されているのでこの探索は成功である。

そこで Prolog は「yes」と答える。このように Prolog は与えられた質問に対して、事実として宣言されているかどうかだけでなく、その答がルールから導出されるかどうかを探索する。

Prolog は導出原理を利用して与えられた課題を遂行する。導出原理は、簡単に言えばある節集合 K に含まれる二つの節 C1 と C2 がそれぞれ相補的なリテラル L1 と L2 を含んでいるとき、K から(C1-{L1})∪(C2-{L2}) を推論して良いという規則である。このとき、(C1-{L1})∪(C2-{L2}) を C1 と C2 の「リゾルヴェント」と呼び、C1・C2 で表わす。例えば{P, Q}と{¬P, R}という二つの節からリゾルヴェント{P, Q}・{¬P, R}={Q, R}が得られる。

ただしこの例は命題論理の言語で書かれているために変数を考慮していない。実際の導出原理は一階述語論理の体系で使われるために、それが適用される条件はかなり複雑なものになる。例えば C1 = {P(x, y), Q(x)}、C2 = {¬P(t(v), z), R(z, w)}という二つの節が含まれている節集合 K に導出原理を適用することを考えよう。C1 に現れる変数 x を t(v) に、C2 に現れる変数 z を変数 y に置き換えたときに両者が相補的なリテラルを含むことになる。そこでそのような置換を行なってからそのリテラルを取り除くと、この二つの節から得られるリゾルヴェントは{Q(t(v)), R(y, w)}となる。

しかし二つの節のリテラルを相補的なリテラルにするような置換の方法は一通りとは限らない。例えば上の置換のほかに、C1 に現れる x を t(v)に、y を定数 a に置き換え、C2 に現れる z を定数 a に置き換えてもやはり相補的なリテラルが得られる。そのとき C1・C2 = {Q(t(v)), R(a, w)}となる。しかし私たちは一つの導出を行なった後も、もとの集合に新しく得られたリゾルヴェントを加えた集合にさらに導出原理を適用することを考えなければならない。たとえば C3 = {¬R(b, w)}という節が節集合 K に含まれていたとする（b は定数であるとする）。このとき後者の置換を行なっていた場合、そのリゾルヴェント{Q(t(v)), R(a, w)}と C3 の間では導出原理を適用することができない。述語 R の第一引数が異なる定数になっているからである。前者の置換を行なっていた場合、{Q(t(v)), R(y, w)}と C3 の間では、変数 y を b に置き換えることで相補的なリテラルを得ることが出来る。このとき{Q(t(v)), R(y, w)}と C3 のリゾルヴェントは{Q(t(v))}になる。

後述する Resolution Theorem が成り立つためには、私たちは可能な限り導出原理が適用できるような置換を考慮しなければならない。ここで重要な役割を果たすのが「最汎統合（most general unification）」という操作である。これは、二つの表現の変数をもっとも一般性の高い仕方で置き換えて一致させるアルゴリズムである。この最汎統合を使って述語論理での導出原理は定義される(Robinson, 1965)。

ある節集合Pに含まれる任意の二つの節から得られるリゾルヴェントをすべてもとのP

に加えて出来る集合をRes(P) によって表わすことにする。また$Res^0(P) = P$、$Res^{n+1}(P) = Res(Res^n(P))$ と定義する。このとき次の定理が成り立つ。

Resolution Theorem　節集合Pが充足不可能であるとき、そしてそのときに限り、あるnについて$\square \in Res^n(P)$ が成り立つ(2)。

導出原理による証明は、分析的反証法（あるいは「タブロー」の方法）の異なる定式化である。従って Resolution Theorem は節形式の述語論理における反駁完全性定理である（節形式の述語論理において、ある節集合が充足可能であることは常に証明できるわけではない）。Prolog が与えられた課題に対して、適切な答を返してくれることを保証しているのは、この定理である（ただしもちろん、プログラムと質問によっては答が出せないこともある。それは Prolog に固有の問題ではなく、一階述語論理の決定性に関する問題である）。

導出原理に従った証明は、かなり手間のかかる過程である。例えば 5 種類ほどの原子文を含む命題論理の定理を、導出原理で証明しようと思うと、証明が数ページに及んでしまうこともある。さらに述語論理の定理となると証明の手間は飛躍的に増大する。しかしロビンソンの論文のタイトルにあるとおり、これは機械が述語論理の定理を証明するのに適するように設計された規則である。機械は、与えられた節集合が充足不可能であるならば必ず、単純な操作によってそのことを有限回のステップで証明することが出来る。自然演繹での証明や公理系での証明にはある程度の慣れやひらめきが必要であるが、導出原理を使った証明はまったくの機械的な手続きによって遂行できる。しかし機械的な過程とはいえ、その過程を効率よく実行するための様々な工夫はロビンソンの論文でもすでに検討されている。実際にコンピューターに実装する際には、このような効率化の工夫は最重要の課題になる。

3. 汎化と帰納論理プログラミング

二つの節 C1 と C2 のリゾルヴェント C3 を導出する操作は演繹的な推論である。この操作を逆にして、与えられた C1 と C3 から C1・C2 = C3 となるような節 C2 を求めようというのが ILP の基本的なアイディアになっている。この C2 は C3 というデータを説明する仮説であると考えられる。この仮説というのがどのようなものであるか例を挙げて考えよう。

```
parent( john, bob).
```

child(bob, john).

という二つの節が与えられたとき、ここから例えば child(X, Y) :- parent(Y, X). という仮説が立てられる。しかしながら与えられたデータから立てられる仮説は無数に存在している。だとすると、私たちはどのような仮説を望ましい仮説とするべきか、ということについて何らかの基準を必要とする。あるいは少なくとも仮説を選択するための方針がなければならない。

プロトキンは最汎統合の逆の操作である「最小汎化（least general generalization）」という操作を考案した。これは変数の置き換えによって二つの節D1 とD2 が得られるような節Dを求めるという操作である。もちろんこのような節D は無限に存在する。プロトキンはそのような節の間に包摂という擬順序関係[3]を導入した。ある節Eが節Fを包摂するというのは、ある置換θが存在して、 Eθ⊆Fが成り立つということである。EがFを包摂するならばEはFを論理的に含意する。従って包摂は論理的な一般性の強さの部分順序である。またこの包摂の関係は束構造を作る[4]。プロトキンは包摂関係によって作られる束構造の中で与えられた二つの節D1 とD2 の上限を求める操作として最小汎化の操作を定義した。プロトキンは同時に「相対最小汎化（relative least general generalization）」の概念を考えた。これは与えられた節集合Kを一つの背景知識として、KとDからD1 とD2 が導出されるような（包摂関係に関して）最小のDを求める操作である(Plotkin, 1969)。

このプロトキンの結果から、マグルトンらは与えられた背景知識の下での逆導出の操作を考案し、そしてこれが ILP を生み出した。逆導出は与えられた背景知識 K と節 C1 と C3 に対して、K と C1・C2 = C3 となるような C2 を求めるという操作である。この C2 が C1 と C3 というデータを K という背景知識の下で一般化した仮説であると考えられる。

汎化の操作と同様、この逆導出の操作も決定的なものにはならない。それどころか与えられたデータから逆導出によって導き出せる仮説は一般に無限に存在する。しかしながら求める節の形式に対して様々な制約を課す（例えば単位節のみを考慮するなど）ことによって、その非決定性を減らすことが出来る(Muggleton, 1992, Muggleton & Buntine, 1992)。これは演繹的な推論規則である導出原理の逆の過程であり、データと背景知識から一般的な仮説を形成する帰納的な推論であると考えられる。

こういった結果が、様々な機械学習システムに利用され、それらの機械学習システムは実際に与えられたデータから「法則」を「発見」する、あるいは新しい関係述語を「発明」することに成功している。たとえば機械学習システム Golem はあるたんぱく質に含まれるアミノ酸の種類（一次構造）から、その三次元的な構造（二次構造）を推測する一つのル

ールを発見している(Gillies, 1996)。

4. 仮説の探索

既に述べたように、与えられたデータから立てられる仮説は無限に存在する。従って私たちが何らかの仮説を採用するときには、何らかの基準に照らしてより良いと思われる仮説を選択している。良い仮説の条件は様々あるが、まず仮説は適度に一般的でなければならない。例えば

parent(john, bob).	parent(john, beth).
parent(paul, mike).	parent(paul, jane).
sibling(bob, beth).	sibling(beth, bob).
sibling(mike, jane).	sibling(jane, mike).

という 8 個の節がデータとして与えられたとしよう。ここから考えられる一つの仮説は sibling(bob, beth) :- parent(X, bob), parent(X, beth) である。しかしこの仮説はデータの一つ —— すなわち sibling(bob, beth) —— しか説明しない。一方 sibling(X, Y) :- parent(Z, X), parent(Z, Y) という仮説は sibling(bob, beth)、sibling(beth, bob)、sibling(mike, jane)、sibling(jane, mike) の 4 個のデータを説明する。しかし同時にこの仮説とデータからは sibling(bob, bob) や sibling(beth, beth) などの余分な事実まで導出されてしまう。従ってこの仮説は明らかに不適当である。

ILP においても、仮説の情報的価値を評価し、そしてより適切な仮説を効率よく探索する様々な戦略が採用されている。仮説の評価と探索の戦略は、新しい仮説の発見のための指針であり、ヒューリスティックとして働くものである。現在、様々な ILP のシステムが実現されているが、どのような応用の場面で、どのようなヒューリスティックを採用するかがその ILP のシステムの成功の鍵であるということが出来る。以下、古川・尾崎・植野(2002)を参照して、ILP の探索戦略のいくつかを紹介しよう。

(1) ボトムアップ／トップダウン探索：ILP は可能な仮説のすべてからなる空間に、なんらかの一般性基準に基づく順序関係（または擬順序関係。たとえば前節で言及した包摂の関係など）から作られる束構造を導入し、その中である探索アルゴリズムに従ってより良い仮説を見つけ出そうとする。あるシステムは一般性の最も低い仮説から始めて少しずつ一般化を広げていくボトムアップ探索を行なう一方で、別なシステムは最初に最も一般性の高い仮説を立てておいてから、徐々に一般性を低くして望ましい帰結だけを導けるよう

にするトップダウン探索を行なう。

(2) 深さ優先／幅優先戦略：ある ILP システムは、それがボトムアップ（トップダウン）探索を行なう際、ある一つの初期仮説から始めて、その仮説に一般化（特殊化）の操作を繰り返し行ない、それ以上一般化（特殊化）できなくなった時点で、他の初期仮説の検討に移る（深さ優先戦略）。この戦略では場合によっては解にたどり着かず、無限ループに陥る可能性がある。別の ILP システムはボトムアップ（トップダウン）探索を行なう際、複数ある初期仮説のすべてに一般化（特殊化）の操作を行なってから、次の段階での一般化（特殊化）の操作に移る（幅優先戦略）。この戦略ではあらゆる仮説を網羅的に探索することになるので、大量のメモリを食うという欠点がある。

(3) 最良優先戦略：ある ILP システムは、仮説の強弱（一般性）に関係なく、その仮説を展開することの好ましさの判断に従って、その時点で最も好ましいと思われる仮説を展開していく。しかしこの探索法では局所最適解に陥ったときに抜け出せないという欠点がある。

(4) バイアス：探索の深さや仮説の空間に何らかの制限を課す戦略もある。このような制限はバイアスと呼ばれる。これには仮説の言語そのものを制限する言語バイアス、述語のモード（節の頭部であるか本体であるか）や引数のタイプ（引数の取りうる領域）を制限するモード宣言、推論の深さに制限を課す（例えば関数のモデルを全域帰納的関数に限る）探索バイアスなどがある。

一般に探索の戦略において、メモリの消費量や探索にかかる時間といった経済性と、最適解が得られる信頼性はトレード・オフの関係にあるといえる。ILP は実用的であることが最重要の要件であり、理論的には最適解が求められても、莫大なメモリや時間を必要とするというような戦略は意味がない。ある課題に対してどの戦略を採用するべきかの判断は、その状況に応じて決定される。

5. ILP のインパクト

ILP は AI の応用の様々な分野で大きな成功を収めている。例えば前述したたんぱく質の一次構造から二次構造を推定する規則の発見(Gillies, 1996)、さらに化合物の分子構造と突然変異誘発性との関係の分析などがある(古川 et al, 2002)。ここからギリースは、ILP などの機械学習システムが科学的方法や論理学についての私たちの理解にとって大きなインパクトを持つものであることを論じている。以下でギリースのこの論点を見ていこう。

まず科学的方法に関して。ILP は科学史上、初めて実際的に成功した帰納論理の体系であり、この ILP の成功、とくに ILP を利用した機械学習システムによる科学的法則の発見

は、科学哲学において支配的だった「機械的な帰納法は科学的方法ではない」という論調に対して明確な反例を突きつけるものである。というのも ILP システムは、完全に機械的な手続きに基づいて帰納的推論を行なうものであり、その意味では帰納的推論の規則を明示化した帰納論理の体系であると考えることができるからである。科学の歴史上初めて、帰納的推論の論理学が体系的に確立され、しかもそれが実際的な科学的成果を上げているという点で、これは画期的な出来事だと評価できる。

次に論理学に関して。ILP システムは、仮説を推論するための規則だけではなく、仮説の評価法、そして仮説の探索戦略を重要なファクターとして持っている。このことからギリースは ILP などの機械学習システムが「新しい論理学の枠組み」を提供するものであると論じている。彼が提案する新しい論理学の枠組みとは「論理＝推論＋制御」というものである。フレーゲ以来の現代論理学は、推論の形式を特徴付けるための理論を確立することに焦点が置かれてきた。しかしそこでは実際に推論をどのように遂行するか、それによって新しい知識をどのように獲得するか、また獲得された知識をどのように評価するかなど、論理学の実践的な側面はほとんど省みられなかった。ILP はそのような現代論理学の趨勢に対して新しい観点を示唆する。

またこの新しい論理学の枠組みを受け入れるならば、論理学はアプリオリな学問であるという一般的な理解に対しても、ILP は異議を唱える。ある ILP システム――その推論の規則、仮説の評価方法、探索戦略――が何らかの意味で正しいものであるかどうかは、そのシステムがどのような実用の場面で、どれだけの成果を挙げられるかということと独立に論じることは出来ない。ここからある論理体系の正しさは、必ずしもアプリオリに決定されるものではないということが帰結する。

20 世紀半ば以降、クワインなどに代表されるように、数学や論理学におけるアプリオリズムを否定する立場が台頭している。彼らが根拠とするのは主として数学的認識あるいは論理学的認識のダイナミズムや、それらがもつ社会性、歴史性である。しかし彼らの念頭にあるのはあくまでも数学者や論理学者のコミュニティーであり、その意味ではやはり数学的認識や論理学的認識は、内部で閉じられたものであるといえる。一方で、ギリースの主張はさらに徹底したアポステリオリズムである。というのもギリースは、論理学のシステムを、単に理論としてではなく、推論を駆動させる制御部を備えたものとして考え、そしてある論理学のシステムの正しさはそれがある分野に応用した際にどれだけ成功するかという基準によって判断されると考えているからである。この論点はそうとうラジカルなものではあるが、機械学習という具体的かつ最新の科学的成果に基づいて主張されている点で興味深い。

機械学習にはILP以外にも様々なアプローチがある。また最近では遺伝的アルゴリズムという、ILPとは全く異なる発想に基づくシステムをILPの中に組み込む試みも行なわれている(市瀬・沼尾, 1999, Tmadoni-Nezhad & Muggleton, 2002)。AIが科学的方法や論理についての私たちの理解に対してインパクトを与えるというギリースの論点を、こういったILP以外の機械学習のシステムに照らして考察することも価値があるだろう。

註

(1) 原子式またはその否定をリテラルという．また原子式を正のリテラル，原子式の否定を負のリテラルという．またある原子式とその否定を，相補的なリテラルと呼ぶ．量化子がどのリテラルよりも左側についている形式を冠頭標準形という．

(2) □は空の節を表わす．通常の論理学で言えばこれは矛盾を表わしている．

(3) 反射性と推移性を持つが，反対称性を持たないような関係を擬順序という．関係 R が反対称性を持つというのは，任意の x, y について xRy かつ yRx ならば x = y が成り立つということである．実際には節集合と擬順序関係は束構造を持たない．相互に包摂しあう節からなる同値類を形成し，包摂関係から誘導される同値類の間の順序関係によって，束構造を導入するのである．

(4) ある順序集合が束であるというのは，任意の元 x, y に関して上限と下限が存在するということである．

文献

Bratoko, I. (1986). *Prolog Programming for Artificial Intelligence*, Addison-Wesley Publishing Company Inc. （2001, 安部憲広訳,『Prolog と AI ① —Prolog への入門』, 近代科学社.）

Kowalski, R. (1979). *Logic for Problem Solving*, Elsevier North Holland, Inc. （1992, 浦昭二監修, 山田眞市・菊池光昭・桑野龍夫訳,『論理による解法』, 培風館.）

古川康一・尾崎知伸・植野研. (2002).『帰納論理プログラミング』, 共立出版.

Gillies, D. (1996). *Artificial Intelligence and Scientific Method*, Oxford University Press.

市瀬龍太郎・沼尾正行(1999).「帰納学習における帰納論理プログラミングと遺伝的プログラミングの統合」,『人工知能学会誌』, *14, no. 2*, (pp. 307-314).

Israel, D. J. (1993). 'The Role(s) of Logic in Artificial Intelligence', in Gabbay, Hogger & Robinson (Eds.), *Handbook of Logic in Artificial Intelligence and Logic Programming, 1*, (pp. 1-29).

Mueller, R. A. & Page, R. L. (1988). *Symbolic Computing with Lisp and Prolog*, Wiley.

Muggleton, S. (1992). 'Inductive Logic Programming', in Muggleton (Ed.), in *Inductive Logic Programming*, Academic Press (pp. 3-27).

Muggleton, S. & Buntine, W. (1992). 'Machine Invention of First-order Predicates by Inverting Resolution', in Muggleton (Ed.), *Inductive Logic Programming*, Academic Press (pp. 261-280).

大原育夫.(1988).『人工知能の基礎知識』, コンピュータサイエンス大学講座 20, 近代科学社.

Plotkin, G. D. (1969). 'A Note on Inductive Generalization', in Meltzer & Michie (Eds.), in *Machine Intelligence, 5*, Edinburgh University Press (pp. 153-163).

Robinson, J. A. (1965). 'A Machine-oriented Logic Based on the Resolution Principle', in *Journal of the Association for Computing Machinery, 12, no. 1*, (pp. 23-41).

Tamaddoni-Nezhad, A. & Muggleton, S. (2002). 'A Genetic Algorithms Approach to ILP', in *Proceedings of the 12th International Conference on ILP*, (pp. 285-300).

〔龍谷大学非常勤講師〕